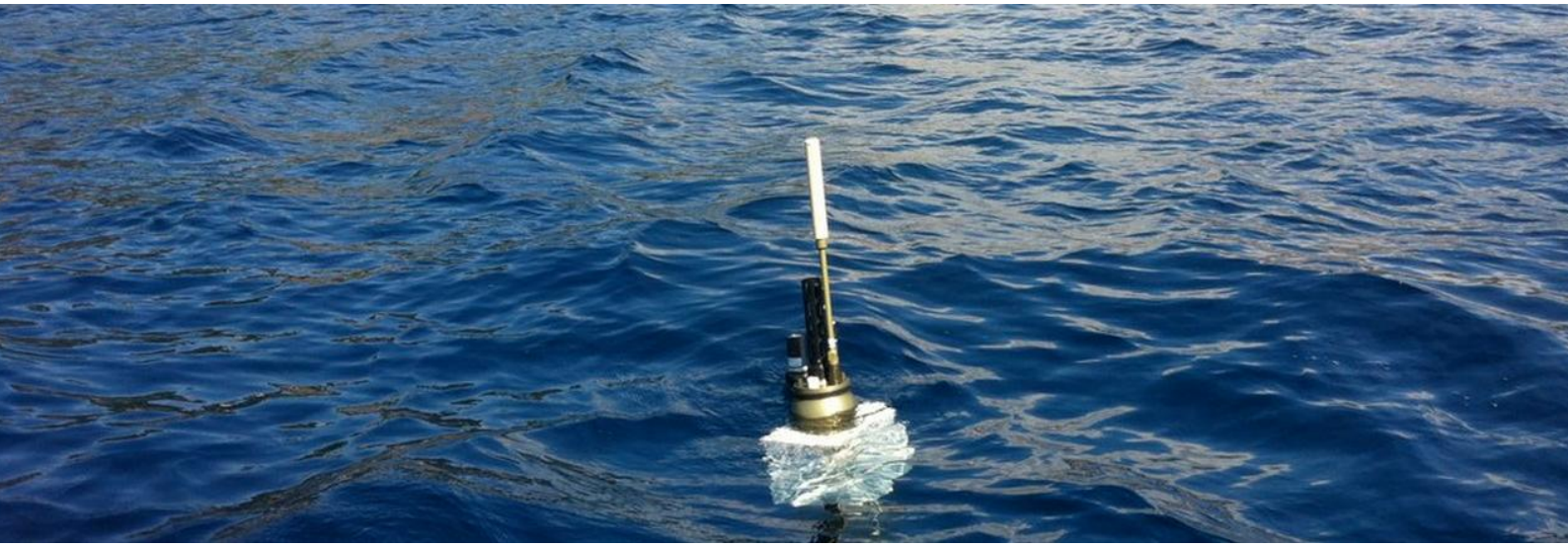
APMT Profiler - File management

AUTOMATED **M**ULTI-**T**ASK **P**ROFILER



33-16-046_APMT_File_Management
Revision 1.4 (2020-01-20)

Table of contents

1. Revision history

Revision	Release date	Notes	Author
1.0	2017-09-04	Original	C. SCHAEFFER
1.1	2018-02-15	Upgrade of technical information: - Adding internal temperature information in [SYSTEM] - Adding pressure activation information in [ACTIVATION] - Adding nke ID information in [USER] - Changing grounding information during navigation phases in [PROFILE] and [ALARM] - Changing feedback information in [ALARM]	C. SCHAEFFER
1.2	2018-11-19	Upgrade of technical information: - Adding pressure reference in “VBatt peak min” tag - Adding USEA identification in [USER] - Adding sensor identification [SENSOR_xxx] - Adding surface sample count in [DATA] - Adding sensor tag in [POWER] - Adding alarms flag in [ALARM] Upgrade of sensor data files: - Adding navigation phase [SURFACE] - Adding decimated raw treatment tag (DW) - Adding a 100 dbar offset in SBE41 pressure - Adding DO, OCR, ECO, C-ROVER, SBEPH, SUNA data format	C. SCHAEFFER
1.3	2019-12-12	Upgrade of technical information: - Adding metadata files - Removing redundant information [USER] - Removing redundant information [SENSOR_xxx] - Changing “kB” into “KB” Upgrade of sensor data files - Adding UVP6 data formats Update SBE41 *.hex standard format: - Adding time stamping for each record Upgrade of technical information: - Adding ice capabilities alarms	C. SCHAEFFER
1.4	2020-01-20	Upgrade of sensor data files - Adding EXTTRIG data format	C. SCHAEFFER

2. File system

Float management is based on a specific file system developed by nke. This file system allows information and data to be stored when the float is turned off, together with the mission configuration.

These files can be accessed by an FTP client via the Bluetooth connection.

2.1 Memory organisation

The memory space is divided into different areas intended for different uses. In particular, the following spaces can be found:

- Reserved: restricted system use to store file manipulation and management information
- Working: used for temporary storage (awaiting transmission).
- Storage (optional): used to expand the temporary storage capacity of the workspace

2.2 File types

The system uses various file types, as needed:

- Configuration file for mission and sensor settings
- Sensor data files (raw, processed and transmitted data)
- Technical data files
- Metadata files (for sensor specific identification)
- Execution trace files (mission progress monitoring)
- Specific files (restricted system use)

2.3 Main memory – Working memory

The working memory (fast access) is itself divided into two physical spaces:

- Area for repeated access files: used for files with recurring access (configuration file...)
- Area for temporary files: used mainly for the generation of sensor information and data during the various navigation phases (a process that automatically deletes the oldest files in favour of new ones allows a minimum free memory space to be maintained)

2.4 Extended memory – Storage memory (optional)

When the application requires it, the use of an additional memory (slow access) allows the storage capacity of the system to be increased.

In this case, the files (awaiting transmission) deleted from the working memory are moved to the extended memory. They will then be repatriated to the working memory (for transmission) when the main memory space is sufficient.

3. Configuration files

The mission configuration is saved as a text file organised by sections and tags.

3.1 Local configuration via Bluetooth

During the mission configuration phase performed locally via Bluetooth, this file can be written and/or modified by the user. It can be manually uploaded in full using an FTP client or partially modified with a TELNET command interpreter.

3.2 Remote configuration via remote commands

When the mission is in progress, the file can be modified by sending remote commands via the satellite communication system (RUDICS...).

3.3 Transmission of the configuration file

The system transmits the current configuration file at various stages of the mission:

- At the start of the mission (initial configuration)
- Upon a change of configuration following the application of a remote command or pre-programmed change (script)

The various files generated are named using the format "xxxx_ccc_mm_apmt.ini", with:

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "mm": pattern number (2 characters)

4. Technical files

The various pieces of technical information associated with a profile are saved in a text file organised by sections.

4.1 Transmission of technical information

The system transmits technical files at various stages of the mission:

- During system verification prior to the mission (self-test)
- During the mission at the end of each pattern
- At end of life

4.1.1. Self-test files

The files generated are named using the format "xxxx_ccc_autotest_nnnnn.txt", with:

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "nnnnn": identification of self-test file number associated with the cycle (5 characters)

4.1.2. Mission files

The files generated are named using the format "xxxx_ccc_mm_technical.txt":

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "mm": pattern number (2 characters)

Note: A pattern numbered "00" corresponds to a pre-mission file.

4.1.3. End-of-life files

The various files generated are named using the format "xxxx_ccc_mm_default_nnnnn.txt", with:

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "mm": pattern number (2 characters)
- "nnnnn": identification of end-of-life file number associated with the cycle/pattern pair (5 characters)

4.2 Technical information available

The data is organised in sections. The various sections and fields that compose them are optional and may not be present in the files.

[SYSTEM]

Information	Format
External pressure offset	Pe offset=xx.xx dbar
Internal pressure	Pi=xxx.x mbar
No-load battery voltage	Vbatt=xx.x V
Battery voltage under load (min.)	Vbatt peak min=xx.x V (xx dbar)
External temperature (air)	Te air=xx.x°C
External pressure (air)	Pe air=xx.x dbar
Internal temperature	Ti=xx.x°C

[GPS]

Information	Format
Time stamping, position (*) and clock drift (**)	UTC=yy-mm-dd hh:mm:ss Lat=ddmm.mmmmh Long=dddmm.mmmmh Clock drift=xx.xxx s

(*) Geographic coordinates are coded in DM format (degrees, minutes and decimal minutes – NMEA standard). For example: “Lat=4747.846N Long=00317.160W” represents 47°47.846’N 3°17.160’W coordinates.

(**) Drift = Float time - GPS time. The offset is reset to 0 after every GPS retiming.

[USER]

Information	Format
nke ID	WC=xxxxxxxxxx

[ACTIVATION]

Information	Format
Pressure activation start	UTC=yy-mm-dd hh:mm:ss Detection start
Pressure activation stop	UTC=yy-mm-dd hh:mm:ss Detection stop

[PROFILE]

Information	Format
Emergence reduction: - Start date/time - Oil volume transferred - Nb. solenoid valve actions	UTC=yy-mm-dd hh:mm:ss Flotation=xxx.x cm3 (xx)
First stabilization - Date/time - Depth	UTC=yy-mm-dd hh:mm:ss First stabilization=xxx dbar
Park descent: - Start date/time - Oil volume transferred - Nb. solenoid valve actions	UTC=yy-mm-dd hh:mm:ss Descent=xxx.x cm3 (xx)
Grounding park descent:	UTC=yy-mm-dd hh:mm:ss Grounding Descent=xxx dbar

- Start date/time - Grounding pressure	
Grounding park descent: - End date/time - Oil volume transferred	UTC=yy-mm-dd hh:mm:ss Grounding Descent escape=xxx.x cm3
Park drift: - Start date/time - Min. depth - Max. depth - Nb. solenoid valve actions - Nb. pump actions - Nb. set point inputs - Nb. set point outputs	UTC=yy-mm-dd hh:mm:ss Park=xxx/xxx dbar (xx/xx) stability=x/x
Stabilized park drift: - Stabilization date/time - Stabilization depth	UTC=yy-mm-dd hh:mm:ss Park stabilization=xxx dbar
Grounding park drift: - Start date/time - Grounding number (1-5) - Grounding pressure	UTC=yy-mm-dd hh:mm:ss Grounding Park x=xxx dbar
Grounding park drift: - End date/time - Grounding number (1-5) - Oil volume transferred	UTC=yy-mm-dd hh:mm:ss Grounding Park x escape=xxx.x cm3
Measurement descent: - Start date/time - Oil volume transferred - Nb. solenoid valve actions	UTC=yy-mm-dd hh:mm:ss Deep profile=xxx.x cm3 (xx)
Grounding measurement descent: - Start date/time - Grounding pressure	UTC=yy-mm-dd hh:mm:ss Grounding Deep profile=xxx dbar
Grounding measurement descent: - End date/time - Oil volume transferred	UTC=yy-mm-dd hh:mm:ss Grounding Deep profile escape=xxx.x cm3
Measurement drift: - Start date/time - Min. depth - Max. depth - Nb. solenoid valve actions - Nb. pump actions - Nb. set point inputs - Nb. set point outputs	UTC=yy-mm-dd hh:mm:ss Short Park=xxx/xxx dbar (xx/xx) stability=x/x
Grounding short park drift: - Start date/time - Grounding number (1-5) - Grounding pressure	UTC=yy-mm-dd hh:mm:ss Grounding Short park x=xxx dbar
Grounding short park drift: - End date/time - Grounding number (1-5) - Oil volume transferred	UTC=yy-mm-dd hh:mm:ss Grounding Short park x escape=xxx.x cm3
Ascent (standard):	UTC=yy-mm-dd hh:mm:ss Ascent=xxx.x cm3 (xx/xx) from xxx

- Start date/time - Oil volume transferred - Nb. pump actions (total) - Nb. actions for take-off - Maximum depth	dbar
Ascent (slow): - Start date/time - Oil volume transferred - Nb. pump actions	UTC=yy-mm-dd hh:mm:ss Ascent (slowly)=xxx.x cm3 (xx)
Ascent (resume): - Start date/time - Oil volume transferred - Nb. pump actions	UTC=yy-mm-dd hh:mm:ss Ascent (resume)=xxx.x cm3 (xx)
Ascent (end) - End date/time	UTC=yy-mm-dd hh:mm:ss Ascent end
Surface - Start date/time	UTC=yy-mm-dd hh:mm:ss Surface
Hanging: - Start date/time - Hanging pressure	UTC=yy-mm-dd hh:mm:ss Hanging=xxx dbar
Hanging: - End date/time	UTC=yy-mm-dd hh:mm:ss Hanging escape
Park drift (ice): - Start date/time - Min. depth - Max. depth - Nb. solenoid valve actions - Nb. pump actions - Nb. set point inputs - Nb. set point outputs	UTC=yy-mm-dd hh:mm:ss Ice Park=xxx/xxx dbar (xx/xx) stability=x/x
Stabilized park drift (ice): - Stabilization date/time - Stabilization depth	UTC=yy-mm-dd hh:mm:ss Ice Park stabilization=xxx dbar
Pressure switch activation	UTC=yy-mm-dd hh:mm:ss Pressure switch activation
Emergency ascent - Start date/time	UTC=yy-mm-dd hh:mm:ss Emergency ascent

Typical rate of descent can be calculated with: “Park drift” UTC - “Emergence reduction” UTC
Typical rate of ascent can be calculated with: “Ascent (end)” UTC – “Ascent (standard)” UTC

[DATA]

Information	Format
Data transmission (*): - Total data size - Nb. files - Average baud rate - Nb. sessions	Upload=xx.x KB of x file(s) at x.x KB/min in x session(s)
Remote command reception: - Nb. accepted	Download=command file (x accepted, x refused, x unknown)

- Nb. refused - Nb. unknown	
Payload configuration download	Download=payload file
Script download	Download=script file
Number of files associated with the pattern (**)	Pattern=x files
Number of points per sensor: - Sensor name [SBE41, DO, OCR, ECO, CROVER, SBEPH, SUNA, UVP6-LPM, UVP6-TX1, UVP6-TX2, UVP6-BLK] - Nb. park descent - Nb. park drift - Nb. measurement descent - Nb. measurement drift - Nb. ascent - Nb. surface - Nb. subsurface (***)	xxxxx=x/x/x/x/x/x points or xxxxx=x/x/x/x/x/x/x points (***)

(*) Information from the previous transmission session.
(**) The following files are not taken into account: technical for self-test or pre-mission, configuration nor command/script acknowledgement. The number of files is expressed prior to the files being broken down for transmission.
(***) SBE41 only

[POWER]

Information	Format
Pattern duration	Pattern=x min
Processing/standby ratio	Processing=xx %
Cumulated hydraulic activations: - Solenoid valve - Pump	SV/Pump=xxx/xxx cs
Cumulated modem activations (*)	Transmission=xxx min
Cumulated GPS activations	GPS=xxx s
Cumulated sensor activations: - Sensor name [SBE41, DO, OCR, ECO, CROVER, SBEPH, SUNA, UVP6] - Duration	xxxx=xxx min

(*) Information from the previous transmission session.

[ALARM]

Information	Format
Deployment	
Start up	Power-on
Invalid configuration	Bad configuration
Excessively heavy float	Flotation (heavy)
Excessively light float	Flotation (light)

Self-test failure: - Source(s) xxx among "FRAM, FLASH, RTC, Vbatt, Pi, Pe, SBE41-Cutoff, SBE41-Offset, GPS, Payload, USEA, Sensor(xxx), Transmitter"	Autotest fail=xxx, …
State	
No-load battery voltage low	Vbatt low
Battery voltage under load low	Vbatt peak low
Low external pressure	Pe low (xxx dbar)
High external pressure	Pe high (xxx dbar)
External pressure fault	Pe default
External breaking pressure	Pe broken
Gear skip	Pe SR high
High internal pressure	Pi high
Presence of water	Water inside
Navigation	
Grounding during park descent	Grounding Descent (xxx dbar)
Grounding during park drift	Grounding Park x (xxx dbar)
Grounding during measurement descent	Grounding Deep profile (xxx dbar)
Grounding during short park drift	Grounding Short park x (xxx dbar)
Hanging during ascent	Hanging (xxx dbar)
Braking during descent	Braking
Operating errors	
System fault	System
Payload board fault	Payload
GPS fault	GPS
Hydraulic fault	Hydraulic
ADC fault	ADC
File fault	File (skip)
RTC fault	RTC
Pressure switch fault	Pressure switch
Ice capabilities	
Ice avoidance (ISA)	Ice (ISA)
Ice avoidance (collision)	Ice (collision)
Ice avoidance (cover)	Ice (cover)
Ice avoidance (ICE period)	Ice (period)
Mode switch	
End of life: - Source xxx among "Pi high, Pe broken, Pe high, Vbatt low, Vbatt peak low, Water inside, Flotation (heavy), Flotation (light)"	End of life (xxx)
Rescue procedure	Rescue
Feedback:	Feedback=xxx (x accepted, x refused)

- Type xxx among "Early profile, Early surface, Abort profile, Abort cycle 0, Abort cycle 1, Go to deep, Standard speed, Lower speed" - Nb. accepted - Nb. refused	Feedback=xxx (x accepted, x refused) …
Sensors	
Bad value in data frame: - Sensor name [SBE41, DO, OCR, ECO, CROVER, SBEPH, SUNA, UVP6] - Error count	xxx value (xxx)
No reply/framing error: - Sensor name [SBE41, DO, OCR, ECO, CROVER, SBEPH, SUNA, UVP6] - Error count	xxx default (xxx)
Repetitive default: - Sensor name [SBE41, DO, OCR, ECO, CROVER, SBEPH, SUNA, UVP6]	xxx broken
Data size: - Sensor name [SBE41, DO, OCR, ECO, CROVER, SBEPH, SUNA, UVP6-LPM, UVP6-TX1, UVP6-TX2, UVP6-BLK] - Data size	xxx size (xxx.x KB)

[SENSOR_SUNA]

Information	Format
Counters: - Sample counter - Power cycle counter - Error counter	Counters=892340/852111/16
Power supply: - Voltage - Intensity	Power supply=9.53 V/0.663 A

[SENSOR_UVP6]

Information	Format
SD memory free space	Available space=xxx.x GB

Example:

[USER]
WC=85E386B7C7
[SYSTEM]
Pe offset=0.00 dbar
Pe air=0.00 dbar
Pi=750.3 mbar
Vbatt=10.7 V
Vbatt peak min=10.0 V (0 dbar)
[GPS]
UTC=18-11-20 12:17:24 Lat=4310.35000S Long=00555.02500W Clock drift=+0.000 s
[PROFILE]
UTC=18-11-20 10:12:22 Flotation=10.7 cm3 (1)
UTC=18-11-20 10:34:03 First stabilization=9 dbar
UTC=18-11-20 10:23:45 Descent=12.2 cm3 (4)
UTC=18-11-20 11:33:06 Park=68/68 dbar (0/0) stability=0/0
UTC=18-11-20 11:33:08 Deep profile=0.0 cm3 (0)
UTC=18-11-20 11:34:06 Short Park=68/68 dbar (0/0) stability=0/0
UTC=18-11-20 11:34:07 Ascent=24.6 cm3 (12/4) from 69 dbar
UTC=18-11-20 12:00:25 Ascent end
UTC=18-11-20 12:10:25 Surface
[DATA]
Upload=4.8 KB of 3 file(s) at 10.2 KB/min in 1 session(s)
Pattern=6 files
SBE41=47/0/0/0/30/6/1 points
OCR=95/0/0/0/33/0 points
ECO=86/0/0/0/30/0 points
SBEPH=58/0/0/0/32/0 points
SUNA=7/0/0/0/4/0 points
[POWER]
Pattern=126 min
Treatment=1 %
EV/Pump=337/382 cs
SBE41=115 min
Transmission=2 min
GPS=260 s
OCR=6 min
ECO=6 min
SBEPH=6 min
SUNA=1 min
[SENSOR_SUNA]
Counters=892340/852111/16
Power supply=9.53 V/0.663 A

5. Metadata files

5.1.Transmission of metadata information

Metadata files are transmitted once at the beginning of the mission.

The files generated are named using the format "xxxx_ccc_mm_metadata.xml", with:

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "mm": pattern number (2 characters)

5.2.Metadata information available

Metadata information are organised in XML format.

[PROFILER]

Information	Format
Identification: - Serial number - Sensor type	<PROFILER SN=”P5XXXX-XXXXXXX” Model=”PROVOR-V”/>

[TELECOM]

Information	Format
Identification: - Type - SIM card ID	<TELECOM Type=”IRIDIUM” CID=” xxxxxxxxxxxxxxx”/>

[HARDWARE]

Information	Format
Control board identification: - Sensor type - Firmware version	<CONTROL_BOARD Model=”APMT” Firmware=”x.xx.xxx”/>
Measure board identification: - Sensor type - Firmware version	<MEASURE_BOARD Model=”USEA” Firmware=”x.xx.xxx”/>

[SENSOR_SBE41]

Information	Format
Identification: - Serial number - Model	<SENSOR SN=”XXXXX” Model=”SBE41-CP”/>
Pressure sensor Identification: - Serial number	<SENSOR_PRESSURE SN="XXXXX”/>

[SENSOR_DO]

Information	Format
Identification: - Serial number - Sensor type	<SENSOR SN=”XXXXX” Model=”DOxxx”/>
Phase correction: - Coefficient “c0”	<PHASE_COEFF c0="xxxx"/>
SVU foil correction: - Coefficient “c0” - ... - Coefficient “c6”	<SVU_FOIL_COEFF c0="xxxx" c1="xxxx" ... c6="xxxx"/>

[SENSOR_OCR]

Information	Format
Identification: - Serial number - Sensor type	<SENSOR SN=”XXXXX” Model=”OCRxxx”/>
Channel correction: - Index [4,7 or 14] - Coefficient “a0” - Coefficient “a1” - Coefficient “im”	<CHANNEL_xx a0=”x.xxxx” a1=”x.xxxx” im=”x.xxxx”/>

[SENSOR_ECO]

Information	Format
Identification: - Serial number - Sensor type	<SENSOR SN=”XXXXX” Model=”ECOxxx”/>
Channel correction: - Index [1-3] - Scale factor “sf” - Dark count “dc”	<CHANNEL_xx sf=”x.xxxx” dc=”xxxxxxx”/>

[SENSOR_CROVER]

Information	Format
Identification: - Serial number	<SENSOR SN=”XXXXX”/>
Path length: - X value “pth”	<PATH_LENGTH pth="xx.xxx"/>
Calibration: - CSCcal value “cln”	<CALIBRATION cln="xxxx"/>

[SENSOR_SBEPH]

Information	Format
Identification: - Serial number	<SENSOR SN=”XXXXX”/>

[SENSOR_SUNA]

Information	Format
Identification: - Serial number - Sensor type - Output pixel begin - Output pixel end	<SENSOR SN=”XXXXX” Model=”SUNA-V2” Spectrum=”Output pixels xx-xx”/>
Sensor board identification: - Firmware version	<SUNA_BOARD Firmware=”x.x.xx”/>
Spectrometer: - Integration time “spintper”	<SPECTROMETER spintper=”xxx”/>

[SENSOR_UVP6]

Information	Format
Identification: - Serial number - Sensor type	<SENSOR SN=”XXXXXXXXXXX” Model=”UVP6-LP”/>
HW configuration	<HW_CONF frame=”xxxxxxxxxx”>
ACQ configuration: - Index [1-10]	<ACQ_CONF_xx frame=”xxxxxxxxxx”>
TAXO configuration	<TAXO_CONF frame=”xxxxxxxxxx”>

Example:

```
<FLOAT>
    <PROFILER SN="AABBCC-DDEEFFF" Model="PROVOR-V"/>
    <TELECOM Type="IRIDIUM" CID="8988169224001045237"/>
    <HARDWARE>
        <CONTROL_BOARD Model="APMT" Firmware="1.07.019"/>
        <MEASURE_BOARD Model="USEA" Firmware="1.00.019"/>
    </HARDWARE>
    <SENSORS>
        <SENSOR_UVP6>
            <SENSOR SN="000001HF" Model="UVP6-LP"/>
            <HW_CONF frame="000001HF,0,ACQ_NKE_00h,0,001,1,150,250,,999.000,393819,10000,2,192.16
            <ACQ_CONF_01 frame="ACQ_NKE_00H,0,1.000,1,1,0,0,1,1,10,0,500,1.0,10,10,0,1000,0,40,la
            <ACQ_CONF_02 frame="ACQ_NKE_00L,0,1.000,1,1,0,0,1,1,10,0,500,1.0,100,10,0,1000,0,40,1
            <ACQ_CONF_03 frame="ACQ_NKE_01H,0,1.000,1,1,0,0,1,1,10,0,500,1.0,10,10,0,1000,0,40,la
            <ACQ_CONF_04 frame="ACQ_NKE_01L,0,1.000,1,1,0,0,1,1,10,0,500,1.0,100,10,0,1000,0,40,1
            <ACQ_CONF_05 frame="ACQ_NKE_20H,0,1.000,1,1,0,0,1,1,10,2,500,1.0,10,10,0,1000,0,40,la
            <ACQ_CONF_06 frame="ACQ_NKE_20L,0,1.000,1,1,0,0,1,1,10,2,500,1.0,100,10,0,1000,0,40,1
            <ACQ_CONF_07 frame="ACQ_NKE_21H,0,1.000,1,1,0,0,1,1,10,2,500,1.0,10,10,0,1000,0,40,la
            <ACQ_CONF_08 frame="ACQ_NKE_21L,0,1.000,1,1,0,0,1,1,10,2,500,1.0,100,10,0,1000,0,40,1
            <ACQ_CONF_09 frame="ACQ_NKE_CUST_1,0,1.000,1,1,0,0,1,1,10,2,500,1.0,15,10,0,1000,0,40
            <ACQ_CONF_10 frame="ACQ_NKE_CUST_2,0,1.000,1,1,0,0,1,1,10,1,500,1.0,30,10,0,1000,0,40
        </SENSOR_UVP6>
        <SENSOR_DO>
            <SENSOR SN="03014" Model="DO4330"/>
            <PHASE_COEFF c0="9.500000e-02"/>
            <SVU_FOIL_COEFF c0="2.749372e-03" c1="1.217100e-04" c2="2.062590e-06" c3="1.655341e+0
        </SENSOR_DO>
        <SENSOR_OCR>
            <SENSOR SN="00184" Model="OCR504"/>
            <CHANNEL_01 a0="2.14766538910e+09" a1="1.63699685504e-07" im="1.161e+00"/>
            <CHANNEL_02 a0="2.14754852610e+09" a1="2.00131764196e-07" im="1.368e+00"/>
            <CHANNEL_03 a0="2.14742382370e+09" a1="2.04019168366e-07" im="1.365e+00"/>
            <CHANNEL_04 a0="2.14745155870e+09" a1="3.03210835119e-06" im="1.359e+00"/>
        </SENSOR_OCR>
        <SENSOR_ECO>
            <SENSOR SN="02345" Model="ECO3"/>
            <CHANNEL_01 sf="7.200e-03" dc="47"/>
            <CHANNEL_02 sf="2.029e-06" dc="43"/>
            <CHANNEL_03 sf="8.720e-02" dc="30"/>
        </SENSOR_ECO>
        <SENSOR_SUNA>
            <SENSOR SN="00555" Model="SUNA-V2" Spectrum="Output pixels 38-77"/>
            <SUNA_BOARD Firmware="2.2.13"/>
            <SPECTROMETER spintper="400"/>
        </SENSOR_SUNA>
        <SENSOR_SBE41>
            <SENSOR SN="08959" Model="SBE41-CP"/>
            <SENSOR_PRESSURE SN="004978241"/>
        </SENSOR_SBE41>
    </SENSORS>
</FLOAT>
```

6. Sensor data files

The system can generate sensor data files under three formats:

- Text format (*.csv)
- Standard binary format (*.hex)
- Extended binary format (*.hex)

6.1. Transmission of sensor data

The various sensor data files generated are named using the format "xxxx_ccc_mm_ssssss.ext", with:

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "mm": pattern number (2 characters)
- "sssss": sensor ID (N characters, e.g. "sbe41")
- "ext": file extension

All data from a sensor are saved in a unique data file associated to the cycle/pattern.

6.2. ID tags

All files are organised in sections using ID tags for the phase and processing type.

6.2.1. Navigation phase identification

At every change of navigation phase, an ID string is inserted. The various ASCII strings are as follows:

- **[DESCENT]**: Data from the descent phase, from the surface to the drift depth
- **[PARK]**: Data from the drift phase
- **[DEEP_PROFILE]**: Data from the descent phase, from the drift depth to the measurement depth
- **[SHORT_PARK]**: Data from the drift phase, from the drift depth to the measurement depth
- **[ASCENT]**: Data from the ascent phase
- **[SURFACE]**: Data from the surface phase

6.2.2. Processing type identification

At every change of phase or processing area, an ID string for the processing type is inserted. The various ASCII strings are as follows:

- **(RW)**: For raw data
- **(DW)**: For decimated raw data
- **(AM)**: For arithmetic mean
- **(SD)**: For standard deviation
- **(MD)**: For median
- **(SS)**: For subsurface point

Several processing types can follow one another. The combinations are as follows:

- (RW)
- (DW)
- (AM)
- (AM)(SD)
- (AM)(MD)
- (AM)(SD)(MD)
- (SS)

6.3.Recording in text format

Spreadsheet format with the possibility of choosing the decimal separator, the column separator as well as the timestamping format can be selected.

```
[DESCENT]
(AM) (SD) (MD)
2015-06-15 08:34:51;4.90;17.4640;35.798;0.0000;0.0000;4.90;17.4640;35.798
2015-06-15 08:34:51;5.50;17.4600;35.797;0.0030;0.0000;5.50;17.4600;35.797
2015-06-15 08:34:51;6.40;17.4530;35.797;0.0030;0.0000;6.40;17.4530;35.797
2015-06-15 08:34:51;7.40;17.4470;35.796;0.0030;0.0000;7.40;17.4500;35.797
2015-06-15 08:34:51;8.50;17.4380;35.796;0.0030;0.0000;8.50;17.4360;35.796
2015-06-15 08:34:51;9.50;17.4320;35.796;0.0030;0.0000;9.50;17.4290;35.796
```

6.4.Recording in binary format

6.4.1. Encoding format

The data is saved in "little endian" format with byte alignment objects:

- **BYTE** - char (1 byte)
- **SHORTINT** - short integer (2 bytes)
- **LONGINT** - long integer (4 bytes)
- **EPOCH** – absolute timestamping (4 bytes) - Unix Epoch format - January 1st, 1970 at 0:00
- **FLOAT** - floating (4 bytes - float) – IEEE-754 format

6.4.2. File structure

Each file begins with an encoding identifying byte:

- 0x01: SBE41 sensor - Extended format
- 0x02: SBE41 sensor - Standard format
- 0x03: DO sensor
- 0x04: OCR504 sensor (4 channels)
- 0x05: OCR507 sensor (7 channels)
- 0x06: OCR507-IR sensor (14 channels)
- 0x07: ECO1 sensor (1 channel)
- 0x08: ECO2 sensor (2 channels)
- 0x09: ECO3 sensor (3 channels)
- 0x0A: CROVER sensor
- 0x0B: SBEPH sensor
- 0x0C: SUNA sensor (45 outputs spectrum)
- 0x0D: SUNA sensor (90 outputs spectrum)
- 0x0E: UVP6 sensor (LPM data)
- 0x0F: UVP6 sensor (TAXO 1 data)
- 0x10: UVP6 sensor (TAXO 2 data)
- 0x11: UVP6 sensor (BLACK data)
- 0x12: ECO1v2 sensor (1 channel)
- 0x13: ECO2v2 sensor (2 channels)
- 0x14: ECO3v2 sensor (3 channels)
- 0x15: ECO4v2 sensor (4 channels)
- 0x16: EXTTRIG sensor

This is followed by groups of recordings identified by the tags.

Example: SBE41 standard format file, descent phase

```
00000000  01 5b 44 45 53 43 45 4e 54 5d 28 44 57 29 cb 65  .[DESCENT](DW)Ëe
00000010  e4 5b 00 00 12 04 c7 57 d6 8b 36 55 00 1e 04 c0  ä[....ÇWÖ‹6U...À
00000020  57 d6 8b 45 56 00 29 04 b2 57 d5 8b 13 80 00 36  WÖ‹EV.).²WÕ‹.€.6
00000030  04 ab 57 d4 8b 62 80 00 41 04 a4 57 d4 8b 61 80  .«WÔ‹b€.A.¤WÔ‹a€
00000040  00 4f 04 9d 57 d4 8b 50 80 00 5d 04 8e 57 d2 8b  .O. WÔ‹P€.].ŽWÒ‹
00000050  38 81 00 68 04 87 57 d1 8b 67 aa 00 74 04 80 57  8 .h.‡WÑ‹gª.t.€W
```

Note: In Iridium RUDICS transmission, files can be supplemented by padding bytes (0x1A). These bytes must not be decoded. Each file can end with a sequence of 0 to 1023 padding bytes.

6.4.3. Timestamping

Records can be timestamped with absolute timestamps (Unix Epoch format) or relative timestamps since a reference date.

Absolute timestamping is used for raw data records in parking phase (where the delay between records can be greater than hours). Data is encoded as follows:

- **Timestamp** absolute encoding (EPOCH)

Relative timestamping is used for the others records. The reference date is saved after the navigation phase tag. The following records use an offset (count in seconds) from this reference time. Data is encoded as follows:

- **Timestamp** offset in seconds (unsigned integer) – code = value - reference time

Example: Relative timestamping encoding

```
00000000  01 5b 44 45 53 43 45 4e 54 5d 28 44 57 29 cb 65  .[DESCENT](DW)Ëe
00000010  e4 5b 00 00 12 04 c7 57 d6 8b 36 55 00 1e 04 c0  ä[....ÇWÖ‹6U...À
00000020  57 d6 8b 45 56 00 29 04 b2 57 d5 8b 13 80 00 36  WÖ‹EV.).²WÕ‹.€.6
00000030  04 ab 57 d4 8b 62 80 00 41 04 a4 57 d4 8b 61 80  .«WÔ‹b€.A.¤WÔ‹a€
00000040  00 4f 04 9d 57 d4 8b 50 80 00 5d 04 8e 57 d2 8b  .O. WÔ‹P€.].ŽWÒ‹
00000050  38 81 00 68 04 87 57 d1 8b 67 aa 00 74 04 80 57  8 .h.‡WÑ‹gª.t.€W
```

Timestamp = 1541694923 (0x5BE465CB) + 0 sec (0x0000) => 2018-11-08 16:35:23

6.5.SBE41 sensor

6.5.1. Data encoding

Data is encoded as follows:

- **Timestamping** of each record
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0
- **Temperature** in m°C and +5.0°C offset (unsigned integer) – code = (value + 5.0) * 1000.0
- **Salinity** in mpsu (unsigned integer) – code = value * 1000.0
- **Temperature standard deviation** in m°C (signed integer) – code = value * 1000.0
- **Salinity standard deviation** in mpsu (signed integer) – code = value * 1000.0

6.5.2. Standard binary format

During the "descent" and "ascent" navigation phases, the first record of each processing area is timestamped in absolute value (reference date), then the recordings for the next records are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all records are timestamped in absolute value.

The recording structures are as follows:

Subsurface - (SS)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiPressure;
  unsigned short int uiATemperature;
  unsigned short int uiSalinity;
 }
 tSSStd;
```

Subsurface data encoding (ascent)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Subsurface record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
				10

Raw data (drift) – (RW) or (DW)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
 }
 tRStdPark;
```

Raw data encoding (park phase)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
				10

Raw data (navigation) – (RW) or (DW)

```
struct
{
 unsigned short int uiDateTimeDelta;
 unsigned short int uiAveragePressure;
 unsigned short int uiAverageTemperature;
 unsigned short int uiAverageSalinity;
}
tRStdNav;
```

Raw data encoding (navigation phase)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
				8

Averaged data – (AM)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
 }
 tMStd;
```

Average data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
				8

Averaged data and standard deviation – (AM)(SD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  signed char cStdDeviationTemperature;
  signed char cStdDeviationSalinity;
 }
 tMECStd;
```

Average + standard deviation data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Standard deviation record				
Temperature STD	BYTE	-0.128 to +0.127°C	0.001°C	1
Salinity STD	BYTE	-0.128 to +0.127 psu	0.001 psu	1
				10

Averaged data and median – (AM)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  unsigned short int uiMedianPressure;
  unsigned short int uiMedianTemperature;
  unsigned short int uiMedianSalinity;
 }
 tMMStd;
```

Average + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
				14

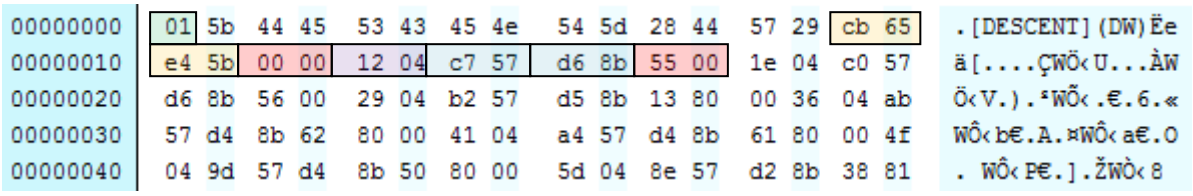
Averaged data, standard deviation and median – (AM)(SD)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  signed char cStdDeviationTemperature;
  signed char cStdDeviationSalinity;
  unsigned short int uiMedianPressure;
  unsigned short int uiMedianTemperature;
  unsigned short int uiMedianSalinity;
 }
 tMECMStd;
```

Average + standard deviation + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Standard deviation record				
Temperature STD	BYTE	-0.128 to +0.127°C	0.001°C	1
Salinity STD	BYTE	-0.128 to +0.127 psu	0.001 psu	1
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
				16

Example:

[DESCENT]
(DW)
2018-11-08 16:35:23,4.2,17.471,35.798
2018-11-08 16:36:48,5.4,17.464,35.798



Timestamp 1 = 1541694923 (0x5BE465CB) + 0 sec (0x0000) => 2018-11-08 16:35:23
Pressure 1 = 1042 (0x0412) => ((1042 / 10.0) – 100.0) = 4.2 dbar
Temperature 1 = 22471 (0x57C7) => ((22471 / 1000.0) – 5.0) = 17.471°C
Salinity 1 = 35798 (0xD68B) => (35798 / 1000.0) = 35.798 psu
Timestamp 2 = Timestamp 1 + 85 sec (0x0055) => 2018-11-08 16:36:48
…

6.5.3. Extended binary format

Advantages of this format compared to the standard version:

- Improved resolution of recordings (pressure and temperature)

During the "descent" and "ascent" navigation phases, the first record of each processing area is timestamped in absolute value (reference date), then the recordings for the next records are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all records are timestamped in absolute value.

The resolution extension for pressure and temperature measurements is encoded on a shared byte using masks, as follows:

- Pressure extension (0.01 dbar) = value & 0xF0
- Temperature extension (0.1 m°C) = value & 0x0F

The available recording structures are as follows:

Subsurface – (SS)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiPressure;
  unsigned short int uiTemperature;
  unsigned short int uiSalinity;
  unsigned char ucPTExtra;
 }
 tSSExt;
```

Subsurface data encoding (ascent)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Subsurface record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Pressure extra Temperature extra	BYTE_H BYTE_L	0.00 to 0.09 dbar 0.0000 to 0.0009°C	0.01 dbar 0.0001°C	1
				11

Raw data (park) – (RW) or (DW)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  unsigned char ucAveragePTExtra;
 }
 tRExtPark;
```

Raw data encoding (park phase)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Pressure extra Temperature extra	BYTE_H BYTE_L	0.00 to 0.09 dbar 0.0000 to 0.0009°C	0.01 dbar 0.0001°C	1
				11

Raw data (navigation) – (RW) or (DW)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  unsigned char ucAveragePTExtra;
 }
 tRExtNav;
```

Raw data encoding (navigation phase)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Pressure extra Temperature extra	BYTE_H BYTE_L	0.00 to 0.09 dbar 0.0000 to 0.0009°C	0.01 dbar 0.0001°C	1
				9

Averaged data – (AM)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  unsigned char ucAveragePTExtra;
 }
 tMExt;
```

Average data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Pressure extra Temperature extra	BYTE_H BYTE_L	0.00 to 0.09 dbar 0.0000 to 0.0009°C	0.01 dbar 0.0001°C	1
				9

Averaged data and standard deviation – (AM)(SD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  unsigned char ucAveragePTExtra;
  signed char cStdDeviationTemperature;
  signed char cStdDeviationSalinity;
 }
 tMECExt;
```

Average + standard deviation data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Pressure extra Temperature extra	BYTE_H BYTE_L	0.00 to 0.09 dbar 0.0000 to 0.0009°C	0.01 dbar 0.0001°C	1

Standard deviation record				
Temperature STD	BYTE	-0.128 to +0.127°C	0.001°C	1
Salinity STD	BYTE	-0.128 to +0.127 psu	0.001 psu	1
				11

Averaged data and median – (AM)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  unsigned char ucAveragePTExtra;
  unsigned short int uiMedianPressure;
  unsigned short int uiMedianTemperature;
  unsigned short int uiMedianSalinity;
  unsigned char ucMedianPTExtra;
 }
 tMMExt;
```

Average + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Pressure extra Temperature extra	BYTE_H BYTE_L	0.00 to 0.09 dbar 0.0000 to 0.0009°C	0.01 dbar 0.0001°C	1
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Pressure extra Temperature extra	BYTE_H BYTE_L	0.00 to 0.09 dbar 0.0000 to 0.0009°C	0.01 dbar 0.0001°C	1
				16

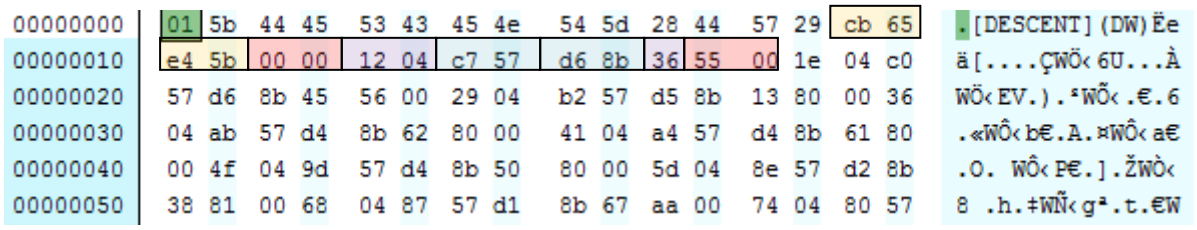
Averaged data, standard deviation and median – (AM)(SD)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  unsigned char ucAveragePTExtra;
  signed char cStdDeviationTemperature;
  signed char cStdDeviationSalinity;
  unsigned short int uiMedianPressure;
  unsigned short int uiMedianTemperature;
  unsigned short int uiMedianSalinity;
  unsigned char ucMedianPTExtra;
 }
 tMECMExt;
```

Average + standard deviation + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Pressure extra Temperature extra	BYTE_H BYTE_L	0.00 to 0.09 dbar 0.0000 to 0.0009°C	0.01 dbar 0.0001°C	1
Standard deviation record				
Temperature STD	BYTE	-0.128 to +0.127°C	0.001°C	1
Salinity STD	BYTE	-0.128 to +0.127 psu	0.001 psu	1
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Pressure extra Temperature extra	BYTE_H BYTE_L	0.00 to 0.09 dbar 0.0000 to 0.0009°C	0.01 dbar 0.0001°C	1
				18

Example:

[DESCENT]
(DW)
2018-11-08 16:35:23,4.23,17.4716,35.798
2018-11-08 16:36:48,5.44,17.4645,35.798



Timestamp 1 = 1541694923 (0x5BE465CB) + 0 sec (0x0000) => 2018-11-08 16:35:23
Pressure 1 = 1042 (0x0412) + 0.03 dbar (0x3_) => ((1042 / 10.0) – 100.0) + 0.03 = 4.23 dbar
Temperature 1 = 22471 (0x57C7) + 0.0006°C (0x_6) => ((22471 / 1000.0) – 5.0) + 0.006 = 17.4716°C
Salinity 1 = 35798 (0xD68B) => (35798 / 1000.0) = 35.798 psu
Timestamp 2 = Timestamp 1 + 85 sec (0x0055) => 2018-11-08 16:36:48
…

6.6.DO sensor

Data is encoded as follows:

- **Timestamping** of each record
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0
- **Phase** in degrees (float) – code = value * 1000.0
- **Temperature** in m°C and +5.0°C offset (unsigned integer) – code = (value + 5.0) * 1000.0
- **Phase standard deviation** in degrees (signed integer) – code = value * 1000.0
- **Temperature standard deviation** in m°C (signed integer) – code = value * 1000.0

During the "descent" and "ascent" navigation phases, the first record of each processing area is timestamped in absolute value (reference date), then the recordings for the next records are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all records are timestamped in absolute value.

The available recording structures are as follows:

Raw data (park) – (RW) or (DW)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiPressure;
  float fC1RPhase;
  float fC2RPhase;
  unsigned short int uiTemperature;
 }
 tRPark;
```

Raw data encoding (park)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
C1 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
C2 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
				16

Raw data (navigation) – (RW) or (DW)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiPressure;
  float fC1RPhase;
  float fC2RPhase;
  unsigned short int uiTemperature;
 }
 tRNav;
```

Raw data encoding (navigation)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
C1 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
C2 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
				14

Averaged data – (AM)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  float fAverageC1RPhase;
  float fAverageC2RPhase;
  unsigned short int uiAverageTemperature;
 }
 tM;
```

Average data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
C1 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
C2 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
				14

Averaged data and standard deviation – (AM)(SD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  float fAverageC1RPhase;
  float fAverageC2RPhase;
  unsigned short int uiAverageTemperature;
  signed short int iStdDeviationC1RPhase;
  signed short int iStdDeviationC2RPhase;
  signed char cStdDeviationTemperature;
 }
 tMEC;
```

Average + standard deviation data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
C1 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
C2 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Standard deviation record				
C1 Phase	SHORTINT	-32.768 to +32.767 deg	0.001 deg	2
C2 Phase	SHORTINT	-32.768 to +32.767 deg	0.001 deg	2
Temperature	BYTE	-0.128 to +0.127°C	0.001°C	1
				19

Averaged data and median – (AM)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  float fAverageC1RPhase;
  float fAverageC2RPhase;
  unsigned short int uiAverageTemperature;
  unsigned short int uiMedianPressure;
  float fMedianC1RPhase;
  float fMedianC2RPhase;
  unsigned short int uiMedianTemperature;
 }
 tMM;
```

Average + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
C1 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
C2 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
C1 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
C2 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
				26

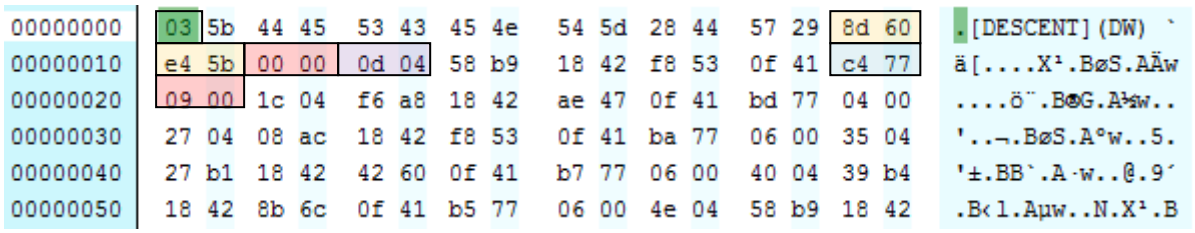
Averaged data, standard deviation and median – (AM)(SD)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  float fAverageC1RPhase;
  float fAverageC2RPhase;
  unsigned short int uiAverageTemperature;
  signed short int iStdDeviationC1RPhase;
  signed short int iStdDeviationC2RPhase;
  signed char cStdDeviationTemperature;
  unsigned short int uiMedianPressure;
  float fMedianC1RPhase;
  float fMedianC2RPhase;
  unsigned short int uiMedianTemperature;
 }
 tMECM;
```

Average + standard deviation + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
C1 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
C2 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Standard deviation record				
C1 Phase	SHORTINT	-32.768 to +32.767 deg	0.001 deg	2
C2 Phase	SHORTINT	-32.768 to +32.767 deg	0.001 deg	2
Temperature	BYTE	-0.128 to +0.127°C	0.001°C	1
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
C1 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
C2 Phase	FLOAT	-90.0 to +90.0 deg	IEEE float	4
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
				31

Example:

[DESCENT]
(DW)
2018-11-08 16:13:01,3.7,38.181,8.958,25.66
2018-11-08 16:13:10,5.2,38.165,8.955,25.65



Timestamp 1 = 1541693581 (0x5BE4608D) + 0 sec (0x0000) => 2018-11-08 16:13:01
Pressure 1 = 1037 (0x040D) => ((1037 / 10.0) – 100.0) = 3.7 dbar
[...]
Temperature 1 = 30660 (0x77C4) => ((30660 / 1000.0) – 5.0) = 25.66°C
Timestamp 2 = Timestamp 1 + 9 sec (0x0009) => 2018-11-08 16:13:10
…

6.7.OCR sensor

Data is encoded as follows:

- **Timestamping** of each record
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0
- **Channel count** (unsigned long int) – code = value
- **Channel count standard deviation** (signed long int) – code = value

Data record can be composed of counts from 1, 2 or 3 channels (depending on the sensor's configuration).

During the "descent" and "ascent" navigation phases, the first record of each processing area is timestamped in absolute value (reference date), then the recordings for the next records are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all records are timestamped in absolute value.

The available recording structures are as follows:

Raw data (park) – (RW) or (DW)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiPressure;
  unsigned long int uliCountChannel1;
  unsigned long int uliCountChannel2;
  [...]
  unsigned long int uliCountChannelN;
 }
 tRPark;
```

4 channels - Raw data encoding (park)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	LONGINT	0 to +4294967295	1	4
Channel 2 count	LONGINT	0 to +4294967295	1	4
Channel 3 count	LONGINT	0 to +4294967295	1	4
Channel 4 count	LONGINT	0 to +4294967295	1	4
				22

Raw data (navigation) – (RW) or (DW)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiPressure;
  unsigned long int uliCountChannel1;
  unsigned long int uliCountChannel2;
  [...]
  unsigned long int uliCountChannelN;
 }
 tRNav;
```

4 channels - Raw data encoding (navigation)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	LONGINT	0 to +4294967295	1	4
Channel 2 count	LONGINT	0 to +4294967295	1	4
Channel 3 count	LONGINT	0 to +4294967295	1	4
Channel 4 count	LONGINT	0 to +4294967295	1	4
				20

Averaged data – (AM)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned long int uliAverageCountChannel1;
  unsigned long int uliAverageCountChannel2;
  [...]
  unsigned long int uliAverageCountChannelN;
 }
 tM;
```

4 channels - Average data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	LONGINT	0 to +4294967295	1	4
Channel 2 count	LONGINT	0 to +4294967295	1	4
Channel 3 count	LONGINT	0 to +4294967295	1	4
Channel 4 count	LONGINT	0 to +4294967295	1	4
				20

Averaged data and standard deviation – (AM)(SD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned long int uliAverageCountChannel1;
  unsigned long int uliAverageCountChannel2;
  [...]
  unsigned long int uliAverageCountChannelN;
  signed long int liStdDeviationCountChannel1;
  signed long int liStdDeviationCountChannel2;
  [...]
  signed long int liStdDeviationCountChannelN;
 }
 tMEC;
```

4 channels - Average + standard deviation data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	LONGINT	0 to +4294967295	1	4
Channel 2 count	LONGINT	0 to +4294967295	1	4
Channel 3 count	LONGINT	0 to +4294967295	1	4
Channel 4 count	LONGINT	0 to +4294967295	1	4
Standard deviation record				
Channel 1 count	LONGINT	-2147483648 to +2147483647	1	4
Channel 2 count	LONGINT	-2147483648 to +2147483647	1	4
Channel 3 count	LONGINT	-2147483648 to +2147483647	1	4
Channel 4 count	LONGINT	-2147483648 to +2147483647	1	4
				34

Averaged data and median – (AM)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned long int uliAverageCountChannel1;
  unsigned long int uliAverageCountChannel2;
  [...]
  unsigned long int uliAverageCountChannelN;
  unsigned short int uiMedianPressure;
  unsigned long int uliMedianCountChannel1;
  unsigned long int uliMedianCountChannel2;
  [...]
  unsigned long int uliMedianCountChannelN;
 }
 tMM;
```

4 channels - Average + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	LONGINT	0 to +4294967295	1	4
Channel 2 count	LONGINT	0 to +4294967295	1	4
Channel 3 count	LONGINT	0 to +4294967295	1	4
Channel 4 count	LONGINT	0 to +4294967295	1	4
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	LONGINT	0 to +4294967295	1	4
Channel 2 count	LONGINT	0 to +4294967295	1	4
Channel 3 count	LONGINT	0 to +4294967295	1	4
Channel 4 count	LONGINT	0 to +4294967295	1	4
				38

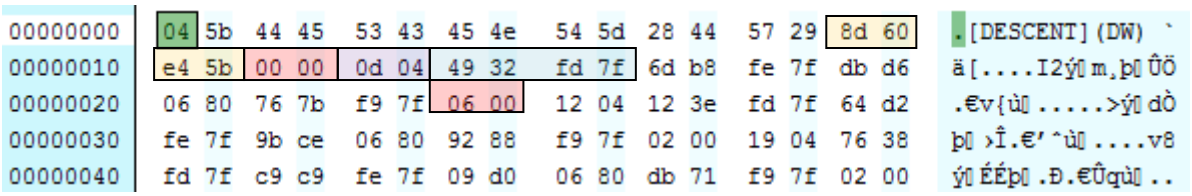
Averaged data, standard deviation and median – (AM)(SD)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned long int uliAverageCountChannel1;
  unsigned long int uliAverageCountChannel2;
  [...]
  unsigned long int uliAverageCountChannelN;
  signed long int liStdDeviationCountChannel1;
  signed long int liStdDeviationCountChannel2;
  [...]
  signed long int liStdDeviationCountChannelN;
  unsigned short int uiMedianPressure;
  unsigned long int uliMedianCountChannel1;
  unsigned long int uliMedianCountChannel2;
  [...]
  unsigned long int uliMedianCountChannelN;
 }
 tMECM;
```

4 channels - Average + standard deviation + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	LONGINT	0 to +4294967295	1	4
Channel 2 count	LONGINT	0 to +4294967295	1	4
Channel 3 count	LONGINT	0 to +4294967295	1	4
Channel 4 count	LONGINT	0 to +4294967295	1	4
Standard deviation record				
Channel 1 count	LONGINT	-2147483648 to +2147483647	1	4
Channel 2 count	LONGINT	-2147483648 to +2147483647	1	4
Channel 3 count	LONGINT	-2147483648 to +2147483647	1	4
Channel 4 count	LONGINT	-2147483648 to +2147483647	1	4
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	LONGINT	0 to +4294967295	1	4
Channel 2 count	LONGINT	0 to +4294967295	1	4
Channel 3 count	LONGINT	0 to +4294967295	1	4
Channel 4 count	LONGINT	0 to +4294967295	1	4
				54

Example:

[DESCENT]
(DW)
2018-11-08 16:13:01,3.7,2147299913,2147399789,2147931867,2147056502
2018-11-08 16:13:07,4.2,2147302930,2147406436,2147929755,2147059858



Timestamp 1 = 1541693581 (0x5BE4608D) + 0 sec (0x0000) => 2018-11-08 16:13:01
Pressure 1 = 1037 (0x040D) => ((1037 / 10.0) – 100.0) = 3.7 dbar
Count 1 = 2147299913 (0x7FFD3249)
[...]
Timestamp 2 = Timestamp 1 + 6 sec (0x0006) => 2018-11-08 16:13:07
…

6.8.ECO sensor

Data is encoded as follows:

- **Timestamping** of each record
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0
- **Channel count** (signed short int) – code = value
- **Channel count standard deviation** (signed char) – code = value

Data record can be composed of counts from 4, 7 or 14 channels (depending on the sensor's configuration).

During the "descent" and "ascent" navigation phases, the first record of each processing area is timestamped in absolute value (reference date), then the recordings for the next records are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all records are timestamped in absolute value.

The available recording structures are as follows:

Raw data (park) – (RW) or (DW)

```
struct
{
 unsigned long int uliDateTime;
 unsigned short int uiPressure;
 signed short int iCountChannel1;
[...]
 signed short int iCountChannelN;
}
 tRPark;
```

3 channels - Raw data encoding (park)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	SHORTINT	-64 to +4130	1	2
Channel 2 count	SHORTINT	-64 to +4130	1	2
Channel 3 count	SHORTINT	-64 to +4130	1	2
				12

Raw data (navigation) – (RW) or (DW)

```
struct
{
 unsigned short int uiDateTimeDelta;
 unsigned short int uiPressure;
 signed short int iCountChannel1;
[...]
 signed short int iCountChannelN;
}
 tRNav;
```

3 channels - Raw data encoding (navigation)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	-	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	SHORTINT	-64 to +4130	1	2
Channel 2 count	SHORTINT	-64 to +4130	1	2
Channel 3 count	SHORTINT	-64 to +4130	1	2
				10

Averaged data – (AM)

```
struct
{
 unsigned short int uiDateTimeDelta;
 unsigned short int uiAveragePressure;
 signed short int iAverageCountChannel1;
[...]
 signed short int iAverageCountChannelN;
}
 tM;
```

3 channels - Average data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	SHORTINT	-64 to +4130	1	2
Channel 2 count	SHORTINT	-64 to +4130	1	2
Channel 3 count	SHORTINT	-64 to +4130	1	2
				10

Averaged data and standard deviation – (AM)(SD)

```
struct
{
 unsigned short int uiDateTimeDelta;
 unsigned short int uiAveragePressure;
 signed short int iAverageCountChannel1;
[...]
 signed short int iAverageCountChannelN;
 signed char cStdDeviationCountChannel1;
[...]
 signed char cStdDeviationCountChannelN;
}
 tMEC;
```

3 channels - Average + standard deviation data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	SHORTINT	-64 to +4130	1	2
Channel 2 count	SHORTINT	-64 to +4130	1	2
Channel 3 count	SHORTINT	-64 to +4130	1	2
Standard deviation record				
Channel 1 count	BYTE	-128 to +127	1	1
Channel 2 count	BYTE	-128 to +127	1	1
Channel 3 count	BYTE	-128 to +127	1	1
				13

Averaged data and median – (AM)(MD)

```
struct
{
 unsigned short int uiDateTimeDelta;
 unsigned short int uiAveragePressure;
 signed short int iAverageCountChannel1;
[...]
 signed short int iAverageCountChannelN;
 unsigned short int uiMedianPressure;
 signed short int iMedianCountChannel1;
[...]
 signed short int iMedianCountChannelN;
}
 tMM;
```

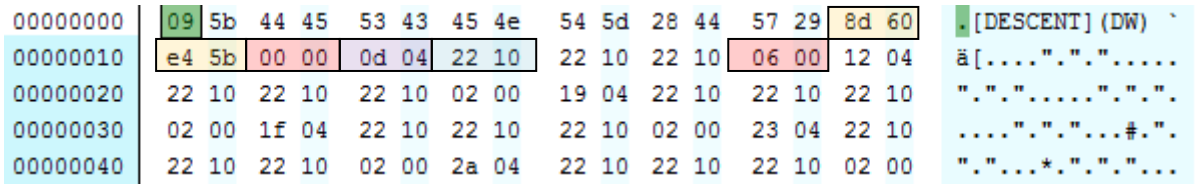
3 channels - Average + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	SHORTINT	-64 to +4130	1	2
Channel 2 count	SHORTINT	-64 to +4130	1	2
Channel 3 count	SHORTINT	-64 to +4130	1	2
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	SHORTINT	-64 to +4130	1	2
Channel 2 count	SHORTINT	-64 to +4130	1	2
Channel 3 count	SHORTINT	-64 to +4130	1	2
				18

Averaged data, standard deviation and median – (AM)(SD)(MD)

```
struct
{
 unsigned short int uiDateTimeDelta;
 unsigned short int uiAveragePressure;
 signed short int iAverageCountChannel1;
 [...]
 signed short int iAverageCountChannelN;
 signed char cStdDeviationCountChannel1;
 [...]
 signed char cStdDeviationCountChannelN;
 unsigned short int uiMedianPressure;
 signed short int iMedianCountChannel1;
 [...]
 signed short int iMedianCountChannelN;
}
 tMECM;
```

3 channels - Average + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	SHORTINT	-64 to +4130	1	2
Channel 2 count	SHORTINT	-64 to +4130	1	2
Channel 3 count	SHORTINT	-64 to +4130	1	2
Standard deviation record				
Channel 1 count	BYTE	-128 to +127	1	1
Channel 2 count	BYTE	-128 to +127	1	1
Channel 3 count	BYTE	-128 to +127	1	1
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Channel 1 count	SHORTINT	-64 to +4130	1	2
Channel 2 count	SHORTINT	-64 to +4130	1	2
Channel 3 count	SHORTINT	-64 to +4130	1	2
				21

Example:
[DESCENT]
(DW)
2018-11-08 16:13:01,3.7,4130,4130,4130
2018-11-08 16:13:07,4.2,4130,4130,4130



Timestamp 1 = 1541693581 (0x5BE4608D) + 0 sec (0x0000) => 2018-11-08 16:13:01
Pressure 1 = 1037 (0x040D) => ((1037 / 10.0) – 100.0) = 3.7 dbar
Count 1 = 4130 (0x1022)
[...]
Timestamp 2 = Timestamp 1 + 6 sec (0x0006) => 2018-11-08 16:13:07
…

6.9.CROVER sensor

Data is encoded as follows:

- **Timestamping** of each record
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0
- **Corrected signal raw count** (signed short int) – code = value
- **Corrected signal raw count standard deviation** (signed short int) – code = value

During the "descent" and "ascent" navigation phases, the first record of each processing area is timestamped in absolute value (reference date), then the recordings for the next records are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all records are timestamped in absolute value.

The available recording structures are as follows:

Raw data (park) – (RW) or (DW)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiPressure;
  signed short int iCorrectedSignalRawCount;
 }
tRPark;
```

Raw data encoding (park)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Corr Sig Raw	SHORTINT	-32768 to +32767	1	2
				8

Raw data (navigation) – (RW) or (DW)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiPressure;
  signed short int iCorrectedSignalRawCount;
 }
tRNav;
```

Raw data encoding (navigation)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Corr Sig Raw	SHORTINT	-32768 to +32767	1	2
				6

Averaged data – (AM)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  signed short int iAverageCorrectedSignalRawCount;
 }
tM;
```

Average data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Corr Sig Raw	SHORTINT	-32768 to +32767	1	2
				6

Averaged data and standard deviation – (AM)(SD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  signed short int iAverageCorrectedSignalRawCount;
  signed short int iStdDeviationCorrectedSignalRawCount;
 }
tMEC;
```

Average + standard deviation data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Corr Sig Raw	SHORTINT	-32768 to +32767	1	2
Standard deviation record				
Corr Sig Raw	SHORTINT	-32768 to +32767	1	2
				8

Averaged data and median – (AM)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  signed short int iAverageCorrectedSignalRawCount;
  unsigned short int uiMedianPressure;
  signed short int iMedianCorrectedSignalRawCount;
 }
tMM;
```

Average + standard deviation data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Corr Sig Raw	FLOAT	-32768 to +32767	1	2
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Corr Sig Raw	SHORTINT	-32768 to +32767	1	2
				10

Averaged data, standard deviation and median – (AM)(SD)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  float fAverageBeamAttenuation;
  signed short int iStdDeviationCorrectedSignalRawCount;
  unsigned short int uiMedianPressure;
  signed short int iMedianCorrectedSignalRawCount;
 }
tMECM;
```

Average + standard deviation data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Corr Sig Raw	SHORTINT	-32768 to +32767	1	2
Standard deviation record				
Corr Sig Raw	SHORTINT	-32768 to +32767	1	2
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Corr Sig Raw	SHORTINT	-32768 to +32767	1	2
				12

6.10. SBEPH sensor

Data is encoded as follows:

- **Timestamping** of each record
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0
- **Voltage** in µV (signed long int) – code = (value * 1000000.0)
- **Voltage standard deviation** in µV (signed char) – code = (value * 1000000.0)

During the "descent" and "ascent" navigation phases, the first record of each processing area is timestamped in absolute value (reference date), then the recordings for the next records are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all records are timestamped in absolute value.

The available recording structures are as follows:

Raw data (park) – (RW) or (DW)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiPressure;
  signed long int liRefVoltage;
 }
tRPark;
```

Raw data encoding (park)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Voltage	LONGINT	-2500000 to +2500000 µV	1 µV	4
				10

Raw data (navigation) – (RW) or (DW)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiPressure;
  signed long int liRefVoltage;
 }
tRNav;
```

Raw data encoding (navigation)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Voltage	LONGINT	-2500000 to +2500000 µV	1 µV	4
				8

Averaged data – (AM)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  signed long int liAverageRefVoltage;
 }
tM;
```

Average data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Voltage	LONGINT	-2500000 to +2500000 µV	1 µV	4
				8

Averaged data and standard deviation – (AM)(SD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  signed long int liAverageRefVoltage;
  signed short int iStdDeviationRefVoltage;
 }
tMEC;
```

Average + standard deviation data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Voltage	LONGINT	-2500000 to +2500000 µV	1 µV	4
Standard deviation record				
Voltage	SHORTINT	-32768 to +32767 µV	1 µV	2
				10

Averaged data and median – (AM)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  signed long int liAverageRefVoltage;
  unsigned short int uiMedianPressure;
  signed long int liMedianRefVoltage;
 }
tMM;
```

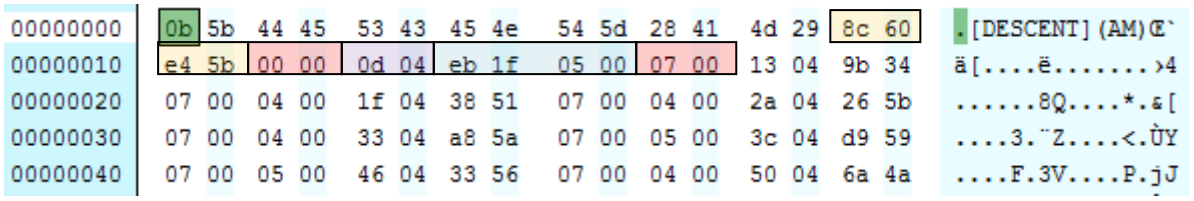
Average + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Voltage	LONGINT	-2500000 to +2500000 µV	1 µV	4
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Voltage	LONGINT	-2500000 to +2500000 µV	1 µV	4
				14

Averaged data, standard deviation and median – (AM)(SD)(MD)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  signed long int liAverageRefVoltage;
  signed short int iStdDeviationRefVoltage;
  unsigned short int uiMedianPressure;
  signed long int liMedianRefVoltage;
 }
tMECM;
```

Average + standard deviation + median data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535 sec	1 sec	2
Average record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Voltage	LONGINT	-2500000 to +2500000 µV	1 µV	4
Standard deviation record				
Voltage	SHORTINT	-32768 to +32767 µV	1 µV	2
Median record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Voltage	LONGINT	-2500000 to +2500000 µV	1 µV	4
				16

Example:
[DESCENT]
(AM)
2018-11-08 16:13:00,3.7,0.335851
2018-11-08 16:13:07,4.3,0.472219



Timestamp 1 = 1541693580 (0x5BE4608C) + 0 sec (0x0000) => 2018-11-08 16:13:00
Pressure 1 = 1037 (0x040D) => ((1037 / 10.0) – 100.0) = 3.7 dbar
Voltage = 335851 (0x00051FEB) => (335851 / 100000.0) = 0.335851 V
Timestamp 2 = Timestamp 1 + 7 sec (0x0007) => 2018-11-08 16:13:07
…

6.11. SUNA sensor

Data is encoded as follows:

- **Timestamping** of each record
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0
- **Temperature** in m°C and +5.0°C offset (unsigned integer) – code = (value + 5.0) * 1000.0
- **Salinity** in mpsu (unsigned integer) – code = value * 1000.0
- **Humidity** in % (unsigned char) – code = (value * 2.0)
- **Dark mean** (unsigned short int) – code = (value * 10.0)
- **Dark standard deviation** (signed short int) – code = (value * 100.0)
- **Nitrate concentration** in µM (float) – code = value
- **Absorbance fit residual RMS** (float) – code = value
- **Output spectrum** (unsigned short int) – code = value

Data record can be composed of output spectrum from 45 or 90 channels (depending on the sensor's configuration).

During the "descent" and "ascent" navigation phases, the first record of each processing area is timestamped in absolute value (reference date), then the recordings for the next records are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all records are timestamped in absolute value.

The available recording structures are as follows:

Raw data (park) – (RW) or (DW)

```
struct
 {
 unsigned long int uliDateTime;
 struct
 {
  struct
  {
   unsigned short int uiPressure;
   unsigned short int uiTemperature;
   unsigned short int uiSalinity;
  }
  tCTD;
  unsigned short int uiInternalTemperature;
  unsigned short int uiSpectrometerTemperature;
  unsigned char ucInternalRelativeHumidity;
  struct
  {
   unsigned short int uiMean;
   signed short int iStandardDeviation;
  }
  tDarkSpectrum;
  struct
```

```
 {
  float fNitrate;
 }
 tSensor;
 float fAbsorbanceFitResiduals;
 struct
 {
  unsigned short int tuiSpectrum[ N ];
 }
 tOutput;
}
tAPFData;
}
tRPark;
```

45 outputs spectrum - Raw data encoding (park)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Temperature int.	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Temperature spec.	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Relative humidity	BYTE	0.0 to +100% RH	0.5% RH	1
Dark M	SHORTINT	0.0 to +6553.5	0.1	2
Dark SD	SHORTINT	-327.68 to +327.67	0.01	2
Nitrate	FLOAT	IEEE float	IEEE float	4
Absorbance	FLOAT	IEEE float	IEEE float	4
Output 1	SHORTINT	0 to +65535	1	2
...	-	-	-	-
Output 45	SHORTINT	0 to +65535	1	2
				117

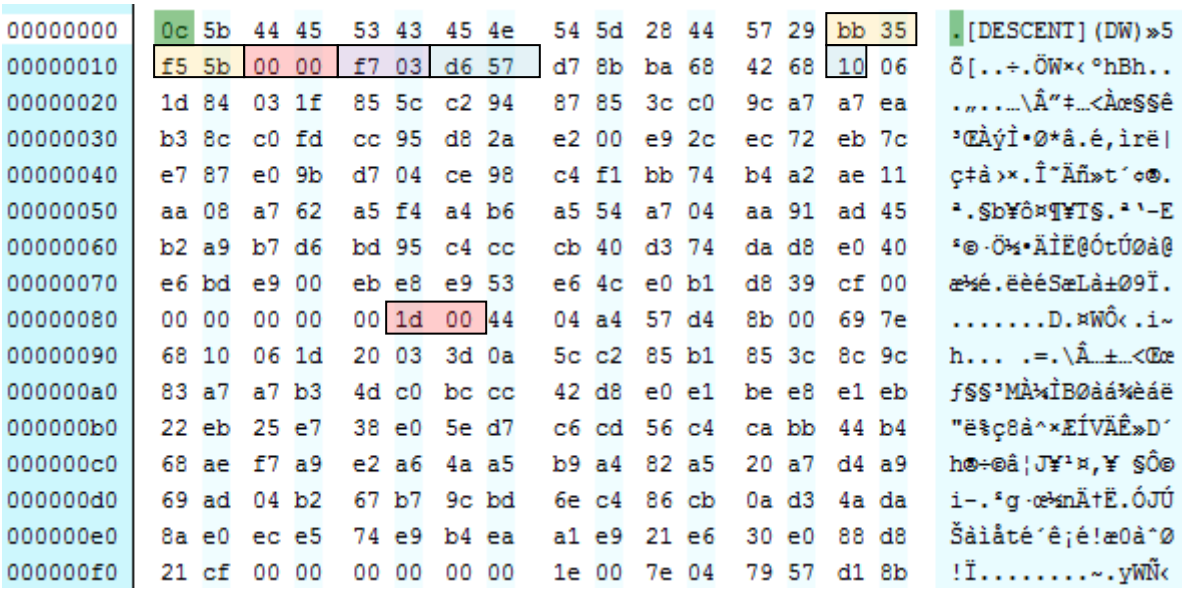
Raw data (navigation) – (RW) or (DW)

```
struct
 {
 unsigned short int uiDateTimeDelta;
 struct
 {
  struct
  {
   unsigned short int uiPressure;
   unsigned short int uiTemperature;
   unsigned short int uiSalinity;
  }
  tCTD;
  unsigned short int uiInternalTemperature;
  unsigned short int uiSpectrometerTemperature;
  unsigned char ucInternalRelativeHumidity;
  struct
  {
   unsigned short int uiMean;
   signed short int iStandardDeviation;
  }
  tDarkSpectrum;
  struct
  {
   float fNitrate;
  }
  tSensor;
  float fAbsorbanceFitResiduals;
  struct
  {
   unsigned short int tuiSpectrum[ N ];
  }
  tOutput;
 }
 tAPFData;
 }
tRNav;
```

45 outputs spectrum - Raw data encoding (navigation)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Salinity	SHORTINT	0.0 to +50.0 psu	0.001 psu	2
Temperature int.	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Temperature spec.	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Relative humidity	BYTE	0.0 to +100% RH	0.5% RH	1
Dark M	SHORTINT	0.0 to +6553.5	0.1	2

Dark SD	SHORTINT	-327.68 to +327.67	0.01	2
Nitrate	FLOAT	IEEE float	IEEE float	4
Absorbance	FLOAT	IEEE float	IEEE float	4
Output 1	SHORTINT	0 to +65535	1	2
...	-	-	-	-
Output 45	SHORTINT	0 to +65535	1	2
				115

Example:
[DESCENT]
(DW)
2018-11-21 10:38:51,1.5,17.486,[...],8.0,[...]
2018-11-21 10:39:20,9.2,[...]



```
00000000  0c 5b 44 45 53 43 45 4e  54 5d 28 44 57 29 bb 35  .[DESCENT](DW)»5
00000010  f5 5b 00 00 f7 03 d6 57  d7 8b ba 68 42 68 10 06  õ[..÷.ÖW×‹ºhBh..
00000020  1d 84 03 1f 85 5c c2 94  87 85 3c c0 9c a7 a7 ea  .„...…\Â”‡…<À§§ê
00000030  b3 8c c0 fd cc 95 d8 2a  e2 00 e9 2c ec 72 eb 7c  ³ŒÀýÌ•Ø*â.é,ìrë|
00000040  e7 87 e0 9b d7 04 ce 98  c4 f1 bb 74 b4 a2 ae 11  ç‡à›×.Î˜Äñ»t´¢®.
00000050  aa 08 a7 62 a5 f4 a4 b6  a5 54 a7 04 aa 91 ad 45  ª.§b¥ô¤¶¥T§.ª'-E
00000060  b2 a9 b7 d6 bd 95 c4 cc  cb 40 d3 74 da d8 e0 40  ²©·Ö½•ÄÌË@ÓtÚØà@
00000070  e6 bd e9 00 eb e8 e9 53  e6 4c e0 b1 d8 39 cf 00  æ½é.ëèéSæLà±Ø9Ï.
00000080  00 00 00 00 00 1d 00 44  04 a4 57 d4 8b 00 69 7e  .......D.¤WÔ‹.i~
00000090  68 10 06 1d 20 03 3d 0a  5c c2 85 b1 85 3c 8c 9c  h... .=.\Â…±…<Œœ
000000a0  83 a7 a7 b3 4d c0 bc cc  42 d8 e0 e1 be e8 e1 eb  ƒ§§³MÀ¼ÌBØàá¾èáë
000000b0  22 eb 25 e7 38 e0 5e d7  c6 cd 56 c4 ca bb 44 b4  "ë%ç8à^×ÆÍVÄÊ»D´
000000c0  68 ae f7 a9 e2 a6 4a a5  b9 a4 82 a5 20 a7 d4 a9  h®÷©â¦J¥¹¤‚¥ §Ô©
000000d0  69 ad 04 b2 67 b7 9c bd  6e c4 86 cb 0a d3 4a da  i-.²g·œ½nÄ†Ë.ÓJÚ
000000e0  8a e0 ec e5 74 e9 b4 ea  a1 e9 21 e6 30 e0 88 d8  Šàìåté´ê¡é!æ0à^Ø
000000f0  21 cf 00 00 00 00 00 00  1e 00 7e 04 79 57 d1 8b  !Ï........~.yWÑ‹
```

Timestamp 1 = 1542796731 (0x5BE435BB) + 0 sec (0x0000) => 2018-11-21 10:38:51
Pressure 1 = 1015 (0x03F7) => ((1015 / 10.0) – 100.0) = 1.5 dbar
Temperature 1 = 22486 (0x57D6) => ((22486 / 1000.0) – 5.0) = 17.486°C
[...]
Humidity 1 = 16 (0x10) => (16 / 2.0) = 8.0 % RH
Timestamp 2 = Timestamp 1 + 29 sec (0x001D) => 2018-11-21 10:39:20
…

6.12. UVP6 sensor (LPM)

Data is encoded as follows:

- **Timestamping** of each record
- **Number of images averaged** in a record – code = value
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0
- **Temperature int.** in m°C and +5.0°C offset (unsigned integer) – code = (value + 5.0) * 1000.0
- **Number of particles per size class** – code = value
- **Mean grey level per size class** (unsigned char) – code = value

During the "descent" and "ascent" navigation phases, the first record of each processing area is timestamped in absolute value (reference date), then the recordings for the next records are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all records are timestamped in absolute value.

Raw data (park) – (RW) or (DW)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned char ucAverageNumber;
  unsigned short int uiPressure;
  unsigned short int uiInternalTemperature;
  float tfObjectCount[ 18 ];
  unsigned char tucGreyLevel[ 18 ];
 }
tRPark;
```

Raw data encoding (park)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Image number	BYTE	1 to +255	1	1
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature int.	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Particles 1 count	FLOAT	IEEE float	IEEE float	4
...	-	-	-	16*4
Particles 18 count	FLOAT	IEEE float	IEEE float	4
Grey level 1	BYTE	0 to 255	1	1
...	-	-	-	16*1
Grey level 18	BYTE	0 to 255	1	1
				99

Raw data (navigation) – (RW) or (DW)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiPressure;
  unsigned short int uiInternalTemperature;
  float tfObjectCount[ 18 ];
  unsigned char tucGreyLevel[ 18 ];
 }
tRNav;
```

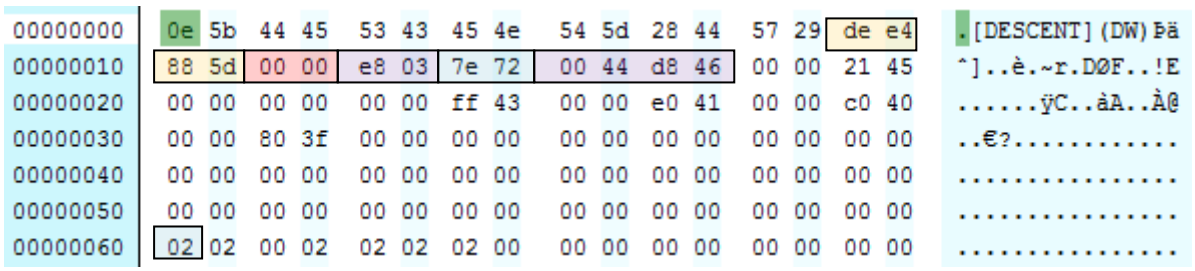
Raw data encoding (navigation)				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535	1 sec	2
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature int.	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Particles 1 count	FLOAT	IEEE float	IEEE float	4
...	-	-	-	16*4
Particles 18 count	FLOAT	IEEE float	IEEE float	4
Grey level 1	BYTE	0 to 255	1	1
...	-	-	-	16*1
Grey level 18	BYTE	0 to 255	1	1
				96

Averaged data – (AM)

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAverageNumber;
  unsigned short int uiAveragePressure;
  unsigned short int uiInternalTemperature;
  float tfObjectCount[ 18 ];
  unsigned char tucGreyLevel[ 18 ];
 }
tM;
```

Average data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	SHORTINT	0 to +65535	1 sec	2
Average record				
Image number	SHORTINT	1 to +65535	1	2
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature int.	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Particles 1 count	FLOAT	IEEE float	IEEE float	4
...	-	-	-	16*4
Particles 18 count	FLOAT	IEEE float	IEEE float	4
Grey level 1	BYTE	0 to 255	1	1
...	-	-	-	16*1
Grey level 18	BYTE	0 to 255	1	1
				98

Example:
[DESCENT]
(DW)
2019-09-23 15:29:34,0,0.00,24.31,27682.0000,2576.0000,0.0000,510.0000,[...],2,2,0,2,2,2,2,0,[...]



Timestamp 1 = 1569252574 (0x5D88E4DE) + 0 sec (0x0000) => 2019-09-23 15:29:34
Pressure 1 = 1015 (0x03E8) => ((1000 / 10.0) – 100.0) = 0.0 dbar
Temperature 1 = 29310 (0x727E) => ((29310 / 1000.0) – 5.0) = 24.310°C
Particle 1/18 1 = 27682.0 (0x46D84400)
[...]
Grey level 1 = 2 (0x02)

6.13. UVP6 sensor (BLACK)

Data is encoded as follows:

- **Timestamping** of each record
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0
- **Internal temperature** in m°C and +5.0°C offset (unsigned integer) – code = (value + 5.0) * 1000.0
- **Number of particles per size class** – code = value

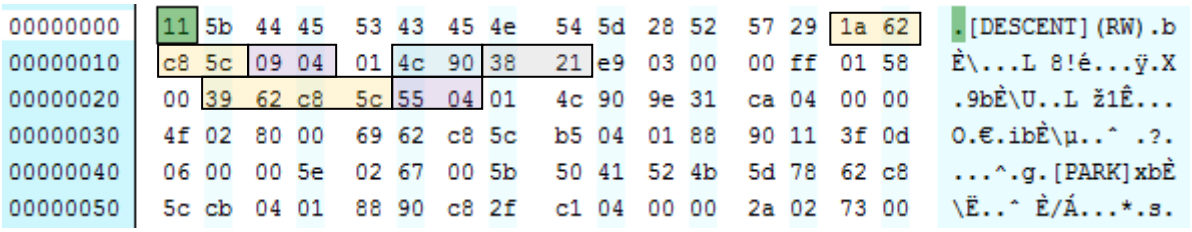
All records are timestamped in absolute value.

Raw data – (RW) or (DW)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiPressure;
  unsigned short int uiInternalTemperature;
  unsigned short int tuiObjectCount[ 5 ];
 }
tR;
```

Raw data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
Temperature int.	SHORTINT	-5.0 to +40.0°C	0.001°C	2
Particles 1 count	SHORTINT	0 to +65535	1	2
...	-	-	-	3*2
Particles 5 count	SHORTINT	0 to +65535	1	2
				18

Example:
[DESCENT]
(RW)
2019-04-30 14:56:26,3.3,1,31.94,8504,1001,0,511,88
2019-04-30 14:56:57,10.9,1,31.94,12702,1226,0,591,128



Timestamp 1 = 1556636188 (0x5CC8621A) => 2019-04-30 14:56:26
Pressure 1 = 1033 (0x0409) => ((1033 / 10.0) – 100.0) = 3.3 dbar
Temperature 1 = 36940 (0x904C) => ((36940 / 1000.0) – 5.0) = 31.94°C
Object count class_1 1 = 8504 (0x2138)
[...]
Timestamp 2 = 1556636217 (0x5CC86239) => 2019-04-30 14:56:57
Pressure 2 = 1109 (0x0455) => ((1109 / 10.0) – 100.0) = 10.9 dbar
...

6.14. EXTTRIG sensor

Data is encoded as follows:

- **Timestamping** of each record
- **Pressure** in cbar and +100 dbar offset (unsigned integer) – code = (value + 100.0) * 10.0

All records are timestamped in absolute value. So, there are no processing type identification strings inserted in the data file.

Raw data – (RW)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiPressure;
 }
tR;
```

Raw data encoding				
Field	**Format**	**Range**	**Resolution**	**Size (Byte)**
Timestamp	EPOCH	-	1 sec	4
Raw record				
Pressure	SHORTINT	-100 to +2500 dbar	0.1 dbar	2
				6

Example:

[DESCENT]
(RW)
2020-01-20 14:08:53,0.0
2020-01-20 14:08:55,0.0

```
00000000  16 5b 44 45 53 43 45 4e 54 5d 75 b4 25 5e e8 03  .[DESCENT]u´%^è.
00000010  77 b4 25 5e e8 03 7a b4 25 5e e8 03 7d b4 25 5e  w´%^è.z´%^è.}´%^
00000020  e8 03 80 b4 25 5e e8 03 83 b4 25 5e e9 03 86 b4  è.€´%^è.ƒ´%^é.†´
00000030  25 5e ea 03 89 b4 25 5e eb 03 8c b4 25 5e ec 03  %^ê.‰´%^ë.Œ´%^ì.
```

Timestamp 1 = 1579529333 (0x5E25B475) => 2020-01-20 14:08:53
Pressure 1 = 1000 (0x03E8) => ((1000 / 10.0) – 100.0) = 0.0 dbar
Timestamp 2 = 1579529335 (0x5E25B477) => 2020-01-20 14:08:55
Pressure 2 = 1000 (0x03E8) => ((1000 / 10.0) – 100.0) = 0.0 dbar
...

nke Instrumentation
6, rue Gutenberg
ZI de Kerandré
56700 Hennebont
FRANCE
Phone: +33 (0)2 97 36 10 12

www.nke-instrumentation.com